Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2011, Article ID 693150, 11 pages
doi:10.1155/2011/693150

Research Article

Low-Power Distributed Kalman Filter for Wireless Sensor Networks

A. Abdelgawad[1] and M. Bayoumi[2]

[1] *The Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70504, USA*
[2] *The Center for Advanced Computer Studies, Department of Computer Science, University of Louisiana at Lafayette, Lafayette, LA 70504, USA*

Correspondence should be addressed to A. Abdelgawad, ama1916@cacs.louisiana.edu

Received 28 April 2010; Revised 30 June 2010; Accepted 2 September 2010

Academic Editor: Xiaorui Wang

Distributed estimation algorithms have attracted a lot of attention in the past few years, particularly in the framework of Wireless Sensor Network (WSN). Distributed Kalman Filter (DKF) is one of the most fundamental distributed estimation algorithms for scalable wireless sensor fusion. Most DKF methods proposed in the literature rely on consensus filters algorithm. The convergence rate of such distributed consensus algorithms typically depends on the network topology. This paper proposes a low-power DKF. The proposed DKF is based on a fast polynomial filter. The idea is to apply a polynomial filter to the network matrix that will shape its spectrum in order to increase the convergence rate by minimizing its second largest eigenvalue. Fast convergence can contribute to significant energy saving. In order to implement the DKF in WSN, more power saving is needed. Since multiplication is the atomic operation of Kalman filter, so saving power at the multiplication level can significantly impact the energy consumption of the DKF. This paper also proposes a novel light-weight and low-power multiplication algorithm. The proposed algorithm aims to decrease the number of instruction cycles, save power, and reduce the memory storage without increasing the code complexity or sacrificing accuracy.

1. Introduction

Wireless sensor network has received momentous attention in recent years because of their titanic potential in applications. Distributed estimation is one of the fundamental problems in WSN. One of the most computationally efficient algorithms for the state estimation is the Kalman filter. Kalman filter is a classical technique with a number of potential distributed applications in WSN. There are several works in the literature that propose distributed Kalman filter based on consensus filter. Consensus filter has proven to be effective tool for performing network distributed computation tasks. Consensus filter allows the network to agree on the value of a particular computation. Consensus filters can be used independently for distributed Kalman filter. The role of this consensus filter is to perform distributed fusion of sensor measurements that is necessary for implementation of a scalable Kalman filter.

Kalman filter form is a basic large class of complex signal processing application. Unlike the other filters algorithms, Kalman filter algorithm does not lend itself for easy implementation; this is because it involves many matrix multiplication, division, and inversion. Altogether, computations of an estimate involve seventeen matrix operations. Among these seventeen matrix operations, there are ten matrix multiplications, two matrix inversions, four matrix additions, and one matrix subtraction. Multiplication is at the core of Kalman filter operations. Moreover, these tasks are computationally intensive and strain the energy resources of any single computational node in a WSN [1]. In other words, most sensor nodes do not have the computational resources to complete multiplication task repeatedly. The sensor nodes available in the market, such as the Crossbow's Micaz [2] and Telosb motes [3], depend on an 8-bit or 16-bit microcontroller. These microcontrollers do not have a floating-point multiplier. To deal with such systems, several

multiplication algorithms have been proposed which rely on repeated additions and consume lots of instruction cycles and exhibit limited precision. Therefore, saving power at the multiplication level has a significant impact on the energy reserve of each node. Consequently, energy-efficient multiplication can extend the WSN's lifetime and increase its computational capabilities. In this paper, we propose a light-weight energy-efficient multiplication algorithm based on Horner's method [4]. Our method aims to reduce the number of add operations during multiplication by rounding any sequence of 1's in the fractional part. The applied rounding reduces the number of instruction cycles, and reduces the memory storage without increasing the code complexity.

Kalman filter is computationally intensive, and it could strain the energy resources of any single computational node in a WSN. Thus, computations will be an issue, the same as the communication, to implement DKF in WSN. The key contribution of the paper is to provide a low-power DKF for WSN. The proposed DKF saves energy in two directions. First, a fast polynomial methodology is proposed to increase the convergence rate of the consensus. Fast convergence can contribute to significant energy saving. Second, a light-weight energy-efficient multiplication algorithm is proposed. The proposed algorithm aims to decrease the number of instruction cycles, save power, and reduce the memory storage without sacrificing the accuracy. Thus, these energy savings make the DKF applicable to be implemented in WSN and hence increase the network's life time.

The rest of the paper is organized as follows: Section 2 introduces the DKF's related work. Some definitions and notions about network representation of the WSN are discussed in Section 3. Central Kalman filter is discussed in Section 4. The consensus filter is introduced at Section 5 to show that the convergence rate of the DKF depends on the magnitude of the second largest eigenvalue. Proposed distributed Kalman filter based on polynomial filter is introduced in Section 6. Simulation results are presented in Section 7. Section 8 introduces the proposed multiplication algorithm. Experimental results are provided in Section 9. Finally, the paper is concluded in Section 10.

2. Related Work

DKF plays an important role in many practical problems connected to sensor networks among which distributed monitoring, tracking, and control. In the recent years, we have witnessed an interest towards this class of problems. Recently studied consensus algorithms, commonly used in the theory of distributed algorithms as an efficient method for data fusion, are providing useful tools to tackle DKF. Consensus problems and their special cases have been the subject of intensive studies by several researchers [5–11]. Low-pass and high-pass consensus filters are also developed to calculate the average of their inputs in sensor networks [12, 13]. Consensus-based tracking and synchronization algorithms in sensor networks that are scalable have recently appeared as powerful tools for mutual information processing [14, 15]. Olfati-Saber in [16] introduced a new DKF algorithm with a peer-to-peer (P2P) architecture that relies on reaching a consensus on estimating of local Kalman filters. The consensus problem with quantized transmission has been studied recently [17]. Schizas et al. in [18] proposed a distributed MLE and BLUE estimators for the estimation of deterministic signals in ad hoc WSNs, where the estimators are formulated as the solution of convex minimization subproblems. Kashyap et al. introduced in [19] the concept of quantized consensus and proposed an algorithm to reach a consensus in that sense. Some contributions found in the literature analyzed the communication bandwidth constrains. Chen and Li studied the tradeoff between bandwidth and tracking accuracy with communication constraints [20]. Ribeiro et al. analyzed the distributed state estimators of dynamical stochastic processes, whereby the low communication cost is affected by requiring the transmission of a single bit per observation [21]. Olfati-Saber introduced a novel distributed Kalman filtering strategy for distributed state estimation and targeted tracking in sensor networks [22, 23]. This DKF strategy consists of identical high-pass consensus filters for distributed fusion of sensor data and covariance information. Speranzon et al. introduced an adaptive strategy for distributed estimation of a time-varying signal measured [24]. A stable filter was derived, as time-varying weights were computed to minimize the estimation error covariance. He discussed the tradeoff between optimality and computational costs. Carli et al. in [25] introduced the problem of estimating the state of a dynamical system from distributed noisy measurements. Spanos et al. introduced a systematic analysis of DKF's performance as various network quantities such as connection density, topology, and bandwidth are varied [26]. Khan and Moura presented a DKF for sparse large-scale systems monitored by sensor networks [27, 28]. Azizi and Khorasani proposed a DKF to estimate actuator faults for deep space formation flying satellites [29].

3. Network Representations

In wireless sensor network, there is a link between two nodes when packets can be successfully delivered from one node to the other. A wireless sensor network is called connected if for two arbitrary nodes, there is a route, which consists of such links, from one to the other. Traditional work on connectivity analysis of wireless sensor networks often focuses on finding a critical transmission range to keep the network connected. However, some low-cost sensor nodes may not support power-adaptive transmissions. On the other hand, changing the transmission range can be reformulated as changing the density of the sensor networks, in which each node is using fixed transmitted power [30]. Recently, random graph theory is introduced into the modeling of sensor networks with uncertain features. A random graph often can be imagined as a living organism which evolves with time. By giving a set of vertices in advance, the edges are generated according to some randomization rules [31].

Let us consider a static network topology. We assume the network at any arbitrary iteration t as an undirected graph $G = \{V, E\}$ with the set of nodes $V = 1, 2, \ldots, n$ and E is the edge set at iteration t. $E \subseteq E^*$, where $E^* \subseteq V \times V$ is employed to show the interaction between the nodes in a network and it is drawn if and only if sensor i can communicate with sensor j. The neighbors of the node i are denoted by the set $\{N_i = j \in V : (i, j) \in E\}$.

4. Central Kalman Filter

In 1960, Kalman presented a recursive solution for the discrete-data linear filtering problem [32]. The Kalman filter, since that time, has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. The Kalman filter is a set of mathematical equations that provides an efficient computational solution to discrete time data filtering problems, in essence removing extraneous noise from a given stream of data. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, It is an optimal estimator in the case of Gaussian uncertainties, and it can do so even when the precise nature of the modeled system is unknown. Moreover, the Kalman filter is the best linear estimator for any other distributions.

Let us consider a sensor network with n sensors that are interconnected via an undirected graph as defined in Section 2. The model of a process can be defined as

$$x_{k+1} = A_k x_k + B_k u_k + w_k, \quad k \geq 0, \tag{1}$$

$$z_k = C_k x_k + v_k, \quad k \geq 0, \tag{2}$$

where $z_k \in \mathbb{R}^{\mathbf{np}}$ represents the vector of p-dimensional measurements obtained via n sensors and w_k and v_k are assumed to be zero-mean white noise processes. The process v_k is called measurement noise and w_k is called process noise. The above equations have several variables: A, B, C are system matrices, k is the time index, x is the system state, u is the input to the system, z is the measurement output, w is the process noise, and v is the measurement noise. Both w and v are zero mean mutually uncorrelated white noises with covariance

$$E\left[w_k \;\; w_k^T\right] = Q_k, \qquad E\left[v_k \;\; v_k^T\right] = R_k. \tag{3}$$

The covariance of the estimation is defined as P_k. Additionally, x_0 is the zero-mean initial state of the process with estimation covariance matrix P_0, and it is assumed to be uncorrelated with u_k and v_k. We describe below the Kalman filter iterations in the information form

$$\begin{aligned} M_k^{-1} &= P_k^{-1} + C_k^T R_k^{-1} C_k, \\ K_k &= M_k C_k^T R_k^{-1}, \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k\left(z_k - C_k \hat{x}_{k|k-1}\right), \\ P_{k+1} &= A_k M_k A_k^T + B_k Q_k B_k^T, \\ \hat{x}_{k+1|k} &= A_k \hat{x}_{k|k} + B_k u_k, \end{aligned} \tag{4}$$

where Q_k is the covariance of the process noise w_k and R_k is covariance of the measurement noise v_k. The calculation of Kalman gain (K) not only depends on the known measurement error covariance R_k but also on the state estimation covariance P_k.

5. Consensus Filter

Consensus problems are widely considered in computer science, and they have a long history in this field. They basically formed the field of distributed computing. Formal study of these types of problems goes back to people who were working in management science and statistics in the 1960s. The notion of statistical consensus theory by DeGroot attracted the interests twenty years later in the problem of processing information with uncertainty obtained from multiple sensors and medical experts [33]. Distributed computing has been considered by people in systems and control theory starting with the work of Tsitsiklis and Athans [34] on asynchronous asymptotic agreement problem for distributed decision-making systems.

In a network, consensus means to get an agreement regarding some common interest of the nodes which depends on the states of all of them. A consensus algorithm is the law which specifies the information flow between a node and its neighbors to reach to the consensus in the whole network [35]. To reach a consensus on a graph, each sensor node i reports a scalar value $x_0(i) \in \mathbb{R}$. The vector of initial values on the network x_0 is denoted by

$$x_0 = [x_0(1), x_0(2), \ldots, x_0(n)]^T \in \mathbb{R}^{\mathbf{n}}. \tag{5}$$

The purpose of the consensus algorithm is to compute the average at each sensor node using linear distribution iteration. Thus, the distributed linear iterations of the network can be defined in the following form:

$$x_{t+1}(i) = W_{ii} x_t(i) + \sum_{j \in N_i} W_{ij} x_t(j), \tag{6}$$

where $i = 1, \ldots, n$, $x_t(j)$ is the value computed by sensor node j at iteration t, and W_{ij} represents the edge weights of G. Each sensor node communicates only with its direct neighbors, so $W_{ij} = 0$. Writing in a matrix-vector format, the above update equation becomes

$$x_{t+1} = W_{ii} x_t, \tag{7}$$

where W_{ii} is the weight matrix corresponding to the graph G of iteration t. The iterative relation given by (7) can be written as

$$x_{t+1} = \left(\prod_{i=0}^{t-1} W^t\right) x_t. \tag{8}$$

Equation (7) means that $x_t = W^t x_0$ for all t. We want to chose the weight matrix W so that for any initial value x_0, x_t convergences to the average vector $\text{Avg} = ((\mathbf{1/n})\mathbf{1}^T x_0)\mathbf{1} = (\mathbf{1/n})\mathbf{1}\mathbf{1}^T x_0$, that is,

$$\lim_{t \to \infty} x_t = \lim_{t \to \infty} W^t x_0 = \left(\frac{1}{n}\right)\mathbf{1}\mathbf{1}^T x_0, \tag{9}$$

where **1** is the vector of ones. This is equivalent to the matrix equation

$$\lim_{t\to\infty} W^t = \left(\frac{1}{n}\right)\mathbf{1}\mathbf{1}^T. \tag{10}$$

Let us define the vector of average as

$$\text{Avg} = \frac{1}{n}\sum_{i=1}^{n} x_0(i). \tag{11}$$

From (10) and (11), we can find

$$\lim_{t\to\infty} x_t = \left(\left(\frac{1}{n}\right)\mathbf{1}\mathbf{1}^T\right)x_0 = \bar{x}\mathbf{1}. \tag{12}$$

The asymptotic convergence factor is defined as

$$r_{\text{asymp}}(W) = \underset{x_0\neq\bar{x}t\to\infty}{\text{SUP}}\lim\left(\frac{||x_t - \text{Avg}||_2}{||x_0 - \text{Avg}||_2}\right)^{1/t}. \tag{13}$$

Equation (10) holds if and only if

$$\begin{aligned}\mathbf{1}^T W &= \mathbf{1}^T,\\ \mathbf{W1} &= \mathbf{1},\\ \rho\left(W - \left(\frac{\mathbf{1}}{\mathbf{n}}\right)\mathbf{1}\mathbf{1}^T\right) &< \mathbf{1},\end{aligned} \tag{14}$$

where $\rho(\cdot)$ is the spectral radius of a matrix. Now, (13) can be written as

$$r_{\text{asymp}}(W) = \rho\left(W - \left(\frac{\mathbf{1}}{\mathbf{n}}\right)\mathbf{1}\mathbf{1}^T\right). \tag{15}$$

Since W is symmetric, so its eigenvalues arranges as $\lambda_1(W) \geq \lambda_2(W)\cdots \geq \lambda_n(W)$. $\lambda_2(W)$, the second largest eigenvalue, is a measure of performance/speed of consensus algorithm [35, 36]. Thus, the convergence rate of (7) depends on the magnitude of the second largest eigenvalue λ_2. Since the DKF relies on consensus filters algorithm, so the convergence rate of the DKF depends on λ_2. The proposed DKF uses a polynomial filter in order to control λ_2 as shown in the next section.

6. Proposed Distributed Kalman Filter

We have described the central Kalman filter in the context of a sensor network with n nodes, where each node observes p various measurements. The process we are describing is m-dimensional processes, that is, $x_k \in \mathbb{R}^{\mathbf{m}}$ and the corresponding white noise vectors have appropriate dimensions matching the z_k and x_k. There are n various sensors; each sensor is m-dimensional, meaning there are m different states associated with each sensor and for each of those states, there are p measurements taken. The states are related to each other by way of matrices A and B. Likewise, z_k is extracted from x_k by means of a linear combination dictated by the matrix C.

In the distributed scenario, we will consider each individual sensor one at a time, the expression for state estimate matrix is placed in the context of a sensor network with n sensors and a topology G that is a connected graph illustrating a process of dimension m using $p \leq m$ sensor estimates. At each iteration k, each sensor node calculates the state estimate using the micro-Kalman filter update equations

$$\begin{aligned}M_k^{-1} &= P_k^{-1} + C_k^T R_k^{-1} C_k,\\ K_k &= M_k C_k^T R_k^{-1},\\ P_{k+1} &= A_k M_k A_k^T + B_k Q_k B_k^T.\end{aligned} \tag{16}$$

The local estimate $\hat{x}_{k|k}^{\text{local}}$ is formed by the predicted regional estimate $\hat{x}_{k|k-1}^{\text{reg}}$ and the local measurement z_k

$$\hat{x}_{k|k}^{\text{local}} = \hat{x}_{k|k-1}^{\text{reg}} + K_k\left(z_k - C_k\hat{x}_{k|k-1}^{\text{reg}}\right). \tag{17}$$

The sensor nodes exchange their estimates over the communication channel and combine the estimates in the neighboring nodes N_i

$$\hat{x}_{k|k}^{\text{reg}} = \sum_{j\in N_i} W_{ij}\hat{x}_{k|k}^{\text{Local}}. \tag{18}$$

W is symmetric and its eigenvalues arranges as $\lambda_1(W) \geq \lambda_2(W)\cdots \geq \lambda_n(W)$. The second largest eigenvalue $\lambda_2(W)$ is a measure of the speed of consensus algorithm. Thus, the convergence rate of DKF depends on the magnitude of the second largest eigenvalue λ_2 as discussed in Section 3. We applied a fast polynomial filter on the spectrum of W in order to impact the magnitude of $\lambda_2(W)$, which mainly drives the convergence rate. In particular, the convergence is faster when the second largest eigenvalue is small. The polynomial filter of degree m that is applied on the spectrum of W is defined as

$$p_m(\lambda) = \propto_0 + \propto_1\lambda + \propto_2\lambda_2 + \cdots + \propto_m\lambda_m. \tag{19}$$

The matrix polynomial is given as

$$p_m(W) = \propto_0 I + \propto_1 W + \propto_2 W_2 + \cdots + \propto_m W_m. \tag{20}$$

$p_m(W)$ is a periodic update of the current sensor node's value with a linear combination of its previous values. Now we can rewrite (18)

$$\hat{x}_{k|k}^{\text{reg}} = \sum_{i=0}^{m} p_i(W)\left(\hat{x}_k^{\text{Local}}\right)_i. \tag{21}$$

Each sensor node typically applies polynomial filter for distributed consensus. The α_m's are computed offline assuming that W is known a priori.

The goal is to find the polynomial that leads to the fastest convergence of linear iteration described in (21), for a given weight matrix W and a certain degree m. The optimal polynomial is the one that minimizes the second largest eigenvalue of W. Therefore, we need to solve an optimization problem where the optimization variables are the $m + 1$ polynomial coefficients $\alpha_0, \alpha_1, \ldots, \alpha_m$ and the

objective function is the spectral radius of $W - (\mathbf{1}/\mathbf{n})\mathbf{1}\mathbf{1}^T$. The following optimization problem needs to be solved:

$$\begin{aligned} &\text{Minimize} \quad \rho\left(\sum_{k=0}^{m}\alpha_k W^k\right), \quad \text{where } \alpha \in \mathbb{R}^{\mathbf{m+1}} \\ &\text{subject to} \quad \left(\sum_{k=0}^{m}\propto_k W^k\right)\mathbf{1} = \mathbf{1}. \end{aligned} \tag{22}$$

The Linear Matrix Inequality (LMI) of (22) is equivalent to a set of m polynomial inequalities in W; that is, the leading principal minors of α must be positive. To solve this optimization problem, the auxiliary variable f will be used to bind the objective function, and then the spectral radius constraint is expressed as a linear matrix inequality (LMI). Thus, the following optimization problem needs to be solved [35]

$$\begin{aligned} &\text{Minimize} \quad f, \quad \text{where } f \in \mathbb{R}^{\mathbf{m+1}} \\ &\text{subject to} \quad -fI \le \sum_{k=0}^{m}\alpha_k W^k - \frac{\mathbf{1}\mathbf{1}^T}{n} \le fI, \\ &\qquad\qquad \left(\sum_{k=0}^{m}\propto_k W^k\right)\mathbf{1} = \mathbf{1}. \end{aligned} \tag{23}$$

Since W is symmetric, $\sum_{k=0}^{m}\alpha_k W^k$ will be symmetric as well. Hence, the constraint $W\mathbf{1} = \mathbf{1}$ is sufficient to ensure that $\mathbf{1}$ will be also a left eigenvector of W. Due to the LMI, the above optimization problem becomes equivalent to a semidefinite program (SDP) [35]. SDP is a special case of cone programming and can be efficiently solved by interior point methods. A matrix polynomial p is applied on the weight matrix W to shape its spectrum in order to increase the convergence rate for the DKF. Since the convergence rate is driven by the second eigenvalue $\lambda_2(W)$, it is then possible to increase the convergence rate by careful design of the polynomial p. The computation of the coefficients of the optimal polynomial is formulated as a semidefinite program that can be efficiently solved. In addition, the sensors are allowed to use their previous estimates, in order to accelerate the convergence rate in a finite number of steps. Although using the previous estimates will exploit the memory of sensors, the memory requirements can be adjusted to the memory constraints imposed by the sensor.

W is calculated according to the fast polynomial consensus introduced above. Then, each sensor node applies polynomial filter for distributed consensus, by implementing Algorithm 1. Each sensor uses its previous estimate, in order to accelerate the convergence rate in a finite number of steps.

Each node predicts the regional estimate $\hat{x}^{\text{reg}}_{k+1|k}$ as follows

$$\hat{x}^{\text{reg}}_{k+1|k} = A\hat{x}^{\text{reg}}_{k|k} + B_k u_k. \tag{24}$$

Figure 1 presents the proposed architecture of m nodes (in another word neighbors) running a micro-Kalman filter as a part of the entire n nodes network. It shows also the communication architecture between the nodes. The advantage of the above micro-Kalman filter is that the state estimates produced are identical to the ones obtained via a central Kalman filter, as we will see in the simulation section. Furthermore, a significant advantage for the distributed implementation is the computational costs of the gain matrices. The central Kalman filter gain K has $O(m^2n)$ elements while the gain $M\mu$ of the micro-Kalman filter has $O(m^2)$ elements. This implies that the implementation of the micro-Kalman filter is more computationally feasible than that of the central Kalman filter, especially for large network. Furthermore, our architecture is scalable in terms of the network size n.

Input: polynomial coefficients $\alpha_0, \alpha_1, \ldots, \alpha_m$, tolerance δ
Output: average estimate $\hat{x}^{\text{reg}}_{k|k}$
While new data exists do
Repeat
  for $i = 1, 2, 3, \ldots, m$
    $\hat{x}^{\text{reg}}_{k|k} = \sum_{i=0}^{m} p_i(W)(\hat{x}^{\text{Local}}_k)_i$
    If $\hat{x}^{\text{reg}}_{k|k} - \hat{x}^{\text{local}}_{k|k} < \delta$
    Exit
    else
    $i = i + 1$
    end if
  end
End while

Algorithm 1

7. Simulation Results

In this section, we provide the output performance of the proposed DKF versus the Central Kalman Filtering (CKF). Consider a network of $n = 100$ sensors that are distributed randomly with a topology shown in Figure 2. The values used for the system defined in (1) are

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{25}$$

In the simulation, a heterogeneous network is proposed. Half of the sensors has one kind of sensors and the other half has another kind of sensors (i.e., each half has different C matrix). The two different C matrices used in (2) are

$$C_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad C_2 = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}. \tag{26}$$

The simulation time is 10 seconds with sampling time $Ts = 0.01$ second and initial value as bellow:

$$\begin{aligned} X_0 &= \begin{bmatrix} 0 & 0 \end{bmatrix}, \\ P_0 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned} \tag{27}$$

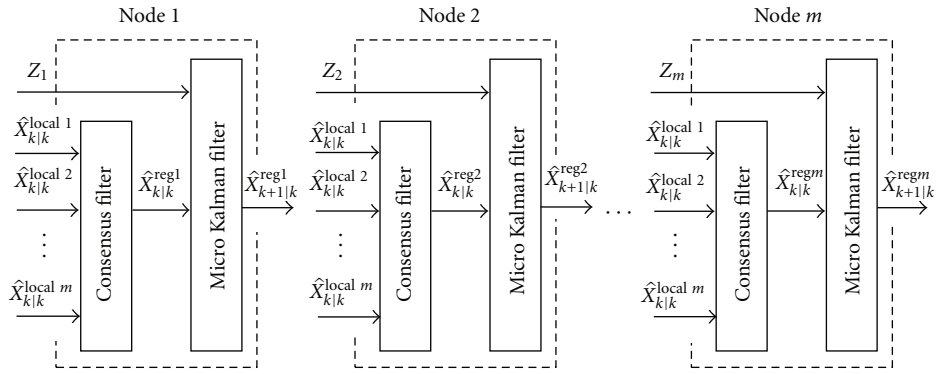

Figure 1: Nodes representation of distributed Kalman filter for m neighbors.

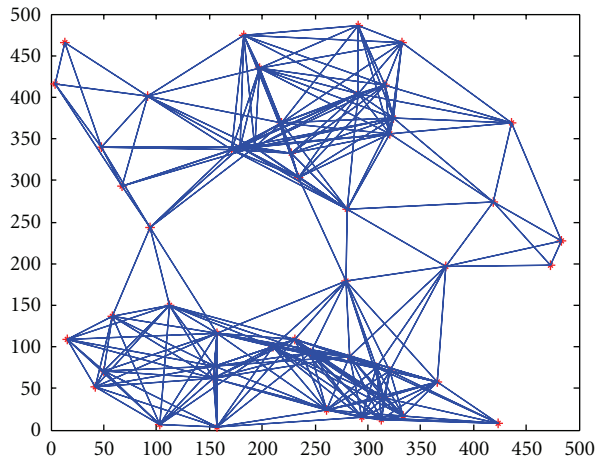

Figure 2: Network topology for $n = 100$ sensor nodes.

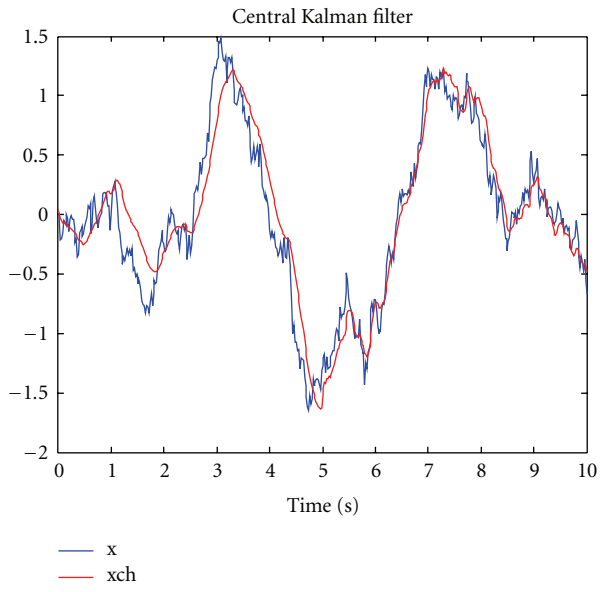

Figure 3: Estimation obtained through the CKF (xch) and the real signal (x).

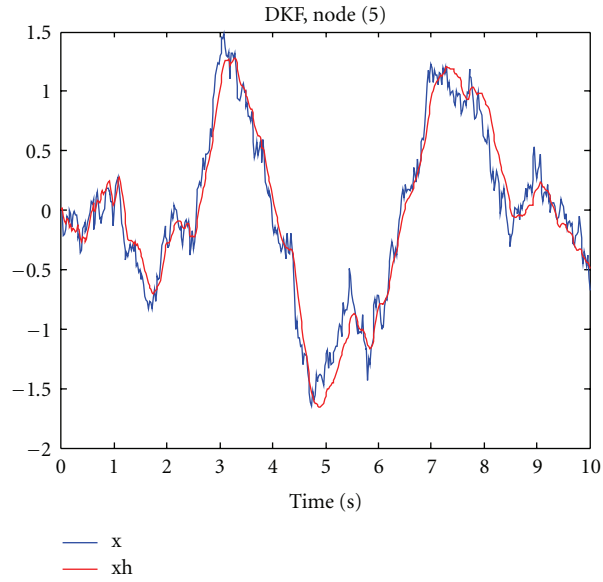

Figure 4: Estimation obtained through DKF (node 5) and the real signal (x).

The estimation obtained from a CKF, shown in Figure 3, will be our reference to evaluate the proposed DKF's performance. Each node in the network has an estimate; Figure 4 shows the squared estimation error for the proposed DKF at node 5 compared with the CKF squared error. Apparently, the proposed distributed and the central Kalman filters provide almost the same estimates and that can be shown clearly in Figure 5 which shows the average Mean Square Error (MSE) for DKF, for all the nodes, versus the MSE of the CKF.

Olfati-Saber in [23] addressed the DKF problem by reducing it into two separate dynamic consensus problems, a low-pass consensus filter for fusion of the measurements and a band-pass consensus filter for fusion of the inverse covariance matrices. He decomposed the central Kalman filter into n micro-Kalman filters with inputs that are provided by two consensus filters. This network of micro-Kalman filters was able to collaboratively provide an estimate

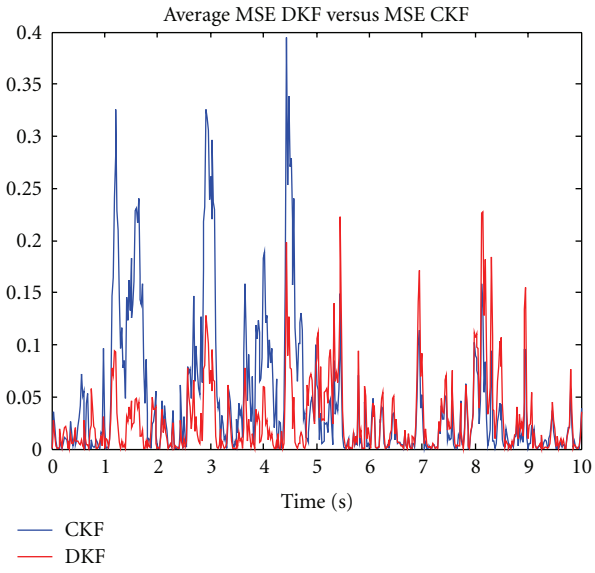

Figure 5: Average MSE for DKF versus MSE for CKF.

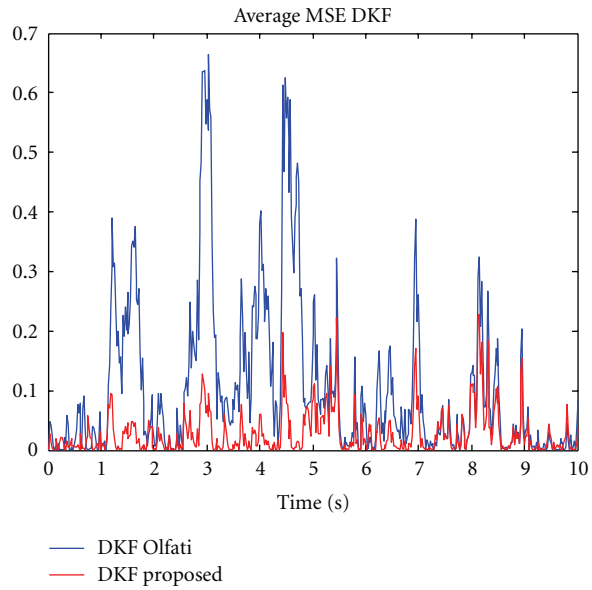

Figure 6: Average MSE for proposed and Olfati's DKF algorithm.

of the state of the observed process. Figure 6 shows a comparison of the performance of both the proposed DKF and Olfati's algorithm. Simulation results are presented for a wireless sensor network with $n = 200$ nodes and 1074 links. The result shows that the proposed algorithm improved the average MSE over the olfatis algorithm.

8. Proposed Multiplication Algorithm

Kalman filter is computationally intensive, and it could strain the energy resources of any single computational node in a WSN. In other words, most sensor nodes do not have the computational resources to complete many of these tasks repeatedly for a long time. Multiplication is at the core of Kalman-filtering operations. Therefore, saving power at the multiplication level will have a significant impact on the energy reserve at each node. Consequently, energy-efficient multiplication can extend the WSN's lifetime and increase its computational capabilities. Most of the sensor nodes available in the market have fixed point microcontroller and do not have hardware multiplier. To deal with such systems, several multiplication algorithms have been proposed which rely on repeated additions and consume lots of instruction cycles and exhibits limited precision. We propose a light-weight energy-efficient multiplication algorithm based on Horner's method [4]. Our method aims to reduce the number of add operations during multiplication by rounding any sequence of 1's in the fractional part. The applied rounding reduces the number of instruction cycles and reduces the memory storage without increasing the code complexity or scarifying accuracy. Moreover, reducing the number of instructions convey to increase the speed of the multiplication and save energy.

8.1. Horner's Method. Horner's method is primarily designed to perform multiplication on devices that do not have a dedicated hardware multiplier. It dictates a set of designed equations, which are unique for any multiplier. These designed equations directly relate to a sequence of shift and add operations on the multiplicand. The Horner's algorithm is based on the positions of the 1's in the multiplier and their distance to the immediate 1 to their left. This is done starting from the rightmost bit position and moving left until the last 1 before the binary point. As an example, consider the multiplication of the two numbers A and B below, represented in 12 bits

$$\begin{aligned} \mathrm{A} &= 0.14325 = 0.001001001010_{\mathrm{b}}, \\ \mathrm{B} &= 0.12345 = 0.000111111001_{\mathrm{b}}. \end{aligned} \tag{28}$$

In the binary equivalent of the multiplier $0.14325 = 0.001001001010_{\mathrm{b}}$, starting from the right the first 1 occurs at bit position 2^{-11}. The difference in position of this 1 to its immediate 1 to the left is two. Similarly, the difference for the 1 in bit position 2^{-9} is three and so on. If the number to be multiplied is denoted as A, the designed equations can be written as follows.

(1) $\mathrm{A1} = \mathrm{A} * 2^{-3} + \mathrm{A}$: set the intermediate result equal to the operand B and start with the rightmost 1. For the first iteration, the weight 2^{-3} is applied to the intermediate result as the distance of the rightmost 1 (bit position 2^{-12}) in the multiplier to its next 1 (bit position 2^{-9}) is three

$$\begin{aligned} &0.000001001001_{\mathrm{b}} + \\ &0.001001001010_{\mathrm{b}} \\ \mathrm{A1} = \quad &\overline{0.001010010011_{\mathrm{b}}}. \end{aligned} \tag{29}$$

(2) $A2 = A1 * 2^{-1} + A$: continue to the next 1 in bit position 2^{-9}. The weight 2^{-1} is now applied to the intermediate result since the distance of the 1 in bit position 2^{-9} to its next 1 (bit position 2^{-8}) is one. The operand is again added

$$\begin{array}{r} 0.000101001001_{b}+ \\ 0.001001001010_{b} \\ \hline 0.001110010011_{b} \end{array} \quad A2 = . \tag{30}$$

(3) $A3 = A2*2^{-1}+A$: keep on to the next 1 in bit position 2^{-7}. The weight 2^{-1} is applied to the intermediate result and the operand added

$$\begin{array}{r} 0.000111001001_{b}+ \\ 0.001001001010_{b} \\ \hline 0.010000010011_{b}. \end{array} \quad A3 = . \tag{31}$$

(4) $A4 = A3 * 2^{-1} + A$: go on to the next 1 in bit position 2^{-6}. The weight 2^{-1} is applied to the intermediate result and the operand added

$$\begin{array}{r} 0.001000001001_{b}+ \\ 0.001001001010_{b} \\ \hline 0.010001010011_{b}. \end{array} \quad A4 = . \tag{32}$$

(5) $A5 = A4 * 2^{-1} + A$: continue to the next 1 in bit position 2^{-5}. The weight 2^{-1} is applied to the intermediate result and the operand added

$$\begin{array}{r} 0.001000101001_{b}+ \\ 0.001001001010_{b} \\ \hline 0.010001110011_{b}. \end{array} \quad A5 = . \tag{33}$$

(6) $A6 = A5*2^{-1}+A$: keep on to the next 1 in bit position 2^{-4}The weight 2^{-1} is applied to the intermediate result and the operand added

$$\begin{array}{r} 0.001000111001_{b}+ \\ 0.001001001010_{b} \\ \hline 0.010010000011_{b}. \end{array} \quad A6 = . \tag{34}$$

(7) The result $= A6 * 2^{-4}$ continue to the last 1 in bit position 2^{-4}. The factor 2^{-4} is applied to the intermediate result, as it is the weight at the position of the leftmost 1. The operand is not added this time, since all the 1's have been taken into account. The result $= A6 * 2^{-4} = 0.000001001000_{b} = 0.017578125$. This has an absolute error of 0.0001060875 which is just 0.434534 LSB, which is 0.60% error from the actual result.

8.2. Proposed Multiplication Method. The proposed method is targeting a fixed-point multiplication by utilizing the redundancy of signed digit code. The feature of redundancy in this representation allows a coefficient implementation to be selected, which in general requires fewer additions and thus yields a faster compact multiplication. The proposed method aims to reduce the number of add operations during multiplication by rounding any sequence of 1's in the fractional part. For example, the number 1010.010111101 becomes 1010.01100001.

Consider the last counter example. The multiplicand B is rounded according to the proposed method to yield B_{new}

$$B_{new} = 0.12345 = 0.001000000001_{b}. \tag{35}$$

The algorithm, then, follows these 2 steps.

(1) $A1 = A * 2^{-9} + A$: set the intermediate result equal to the operand B_{new} and start with the rightmost 1. For the first iteration, the weight 2^{-9} is applied to the intermediate result as the distance of the rightmost 1 (bit position 2^{-12}) in the multiplier to its next 1 (bit position 2^{-3}) is three

$$\begin{array}{r} 0.000000000001_{b}+ \\ 0.001001001010_{b} \\ \hline 0.001001001011_{b}. \end{array} \quad A1 = . \tag{36}$$

(2) The result $= A1 * 2^{-3}$: Proceed to the last 1 in bit position 2^{-3}. The factor 2^{-3} is applied to the intermediate result, as it is the weight at the position of the leftmost 1. The operand is not added this time, since all the 1's have been taken into account. The result $= A1 * 2^{-3} = 0.000001001001_{b} = 0.017822265625$. This has an absolute error of 0.000138053125 which is just 0.5654656 LSB, which is 0.78% error from the actual result. The procedure remains the same if the operand is a negative fraction.

The proposed algorithm takes only 2 steps compared to seven steps for Horner's method. The error of the multiplication comes from the fraction part and depends on the total number of bits in the fractional part. In order to compare the accuracy of the proposed method with the Horner's method, we calculated the average absolute error of multiplying all the possible combination of two fractions for different fraction width (starting from 2 bits to 12 bits). Figure 7 shows the average absolute error for both methods. The simulation results show that the proposed method reduces the accuracy by a maximum of 1% compared to Horner's method. Figure 8 draws a Box and Whisker diagram to show the spread of the absolute error of the proposed multiplication method.

Table 1 shows the comparison of accuracy, memory requirements, speed, and energy for both methods for the same multiplication example we gave before. The proposed method reduces the number of instruction cycles and the code size, which will lead to increase the multiplication speed, without scarifying the accuracy. Experimental results

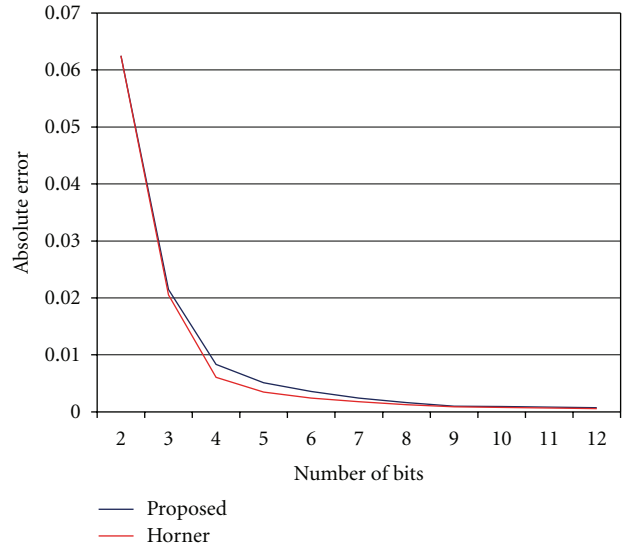

Figure 7: Absolute average multiplication error for both methods.

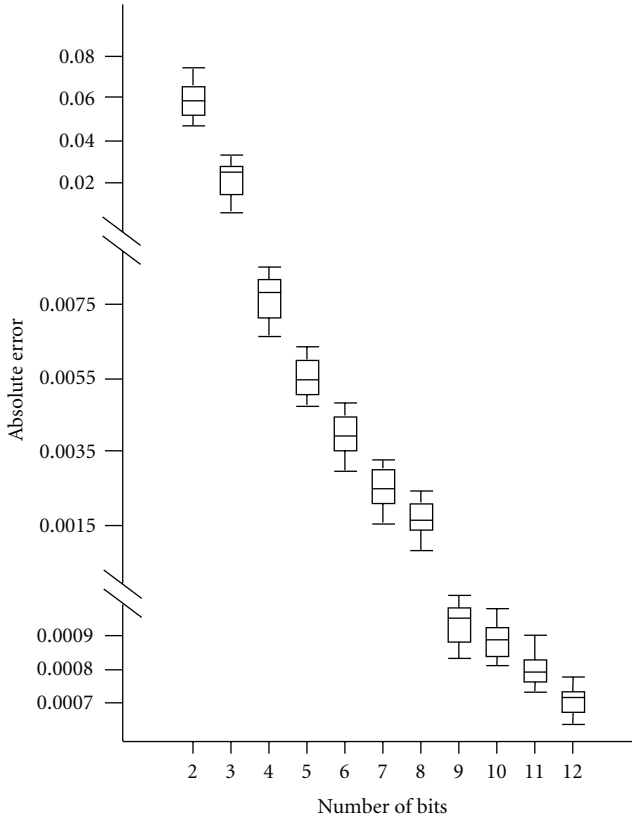

Figure 8: Box and Whisker diagram of the proposed multiplication error.

show that the proposed algorithm achieves up to 17% power saving and 16% increasing in speed, with only maximum 1% accuracy loss compared to Horner's algorithm. The new multiplication method has been validated experimentally using the eZ430-RF2500 wireless sensor board. The experimental results emphasize the simplicity and novelty of this scheme in the use of low power DKF for limited-resource WSN.

Table 1: Comparison for both methods.

Multiplication of $0.14325 * 0.12345$		
	Horner	Proposed
Instruction cycle	18	6
Code size (Byte)	37	26
Output result	0.0175781	0.0178222
Absolute error	0.0001060	0.00013805
Energy (pJ)	0.228121	0.178196
Speed (uS)	0.0218	0.0202

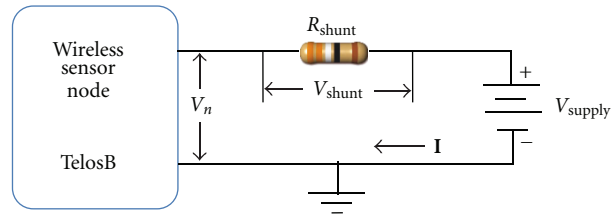

Figure 9: Power measurement with shunt resistor.

9. Experimental Results

A test bed composed from 10 wireless sensor motes, TelosB, was used to test the proposed DKF and measure its power consumption. TelosB is designed for low-power operation. The low power operation of the TelosB module is due to the ultra-low-power Texas Instruments MSP430 F1611 microcontroller featuring 10 kB of RAM, 48 kB of flash, and 128 B of information storage. It supports several low-power operating modes and consumes as low as 1 μA in a standby mode; it also has very fast wake up time of no more than 6 μs. TelosB features a Chipcon 2420 radio in the 2.4 GHz band. The CC2240 is controlled by the MSP430 microcontroller through the SPI port and a series of digital I/O lines with interrupt capabilities. The MAC protocol used is X-MAC. X-MAC is an asynchronous MAC protocol in which the sender uses short preambles to awaken the receiver. Before any transmission, the sender senses the channel, if it is busy the sender retries after a random backoff; otherwise, it sends short preambles embedding the address of the receiver. Once the receiver detects its address, it sends an acknowledgment, and the sender can start transmitting the data [37].

Energy consumption in each TelosB can be attributed to the current draw of each node. Therefore, we can use accurate measurements of the amount of current that the node sinks to determine the power consumption. Current measurement is typically done with a shunt resistor placed in series with the current flow in a circuit as shown in Figure 9. This resistor is specifically chosen to be high precision and low impedance so as not to interfere greatly with the circuit being monitored. Because the value of the resistor is known, by measuring the voltage drop across the shunt resistor, we can accurately calculate the current using Ohm's law as follows.

$$P = V_n * I = \left(V_{\text{supply}} - V_{\text{shunt}}\right) * \frac{V_{\text{shunt}}}{R_{\text{shunt}}}. \quad (37)$$

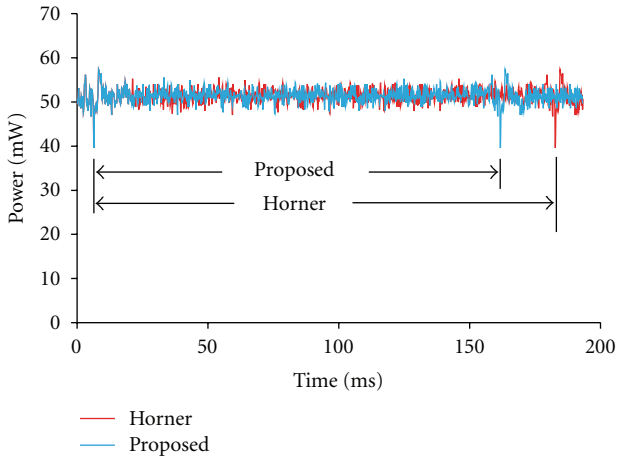

Figure 10: Power traces for DKF using proposed and Horner multiplication methods.

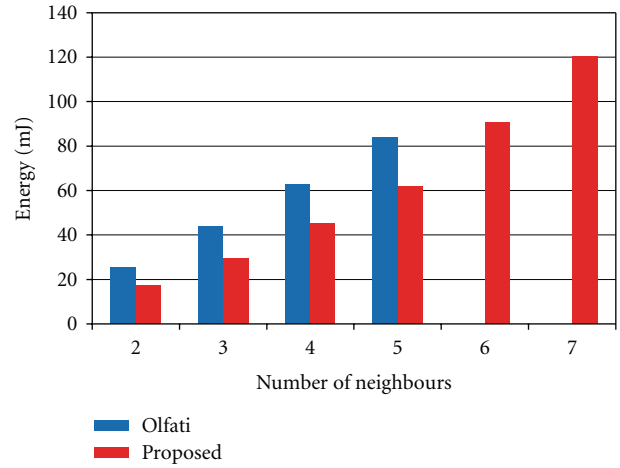

Figure 11: Energy consumption of the proposed DKF and Olfatis' DKF.

Table 2: Energy and time for the proposed polynomial filter.

	Proposed polynomial	Standard polynomial
Energy (mJ)	62.01644	71.05673
Time (S)	14.6193	16.5402

The code for all the nodes was written in NesC. A java GUI was developed to provide a friendly user interface with the nodes. The java interface is a multithreaded socket-based program that communicates with a serial forwarding program. A set of 10 nodes is distributed around the laboratory, they all run a DKF with the same equation values defined in the simulation section, and a laptop gateway is configured to be able to send/receive control signals and data packets to/from the nodes.

To illustrate the effects of energy saving for the proposed multiplication method on the DKF, Figure 10 shows the comparison between the power consumption of one node has 5 neighbors and run DKF using the proposed multiplication method and Horner's method. Figure 10 shows the power trace for only one iteration. The proposed method takes 140 ms, while Horner's method takes 153 ms. Thus, using the proposed multiplication method in DKF saves 8% of energy.

The proposed polynomial filter increases the convergence rate of the DKF. Fast convergence can contribute to significant energy saving and hence a fast DKF. Table 2 shows the time and energy consumption for the DKF using the standard polynomial and the proposed polynomial. The measurements are for one node has five neighbors and runs for ten iterations.

Experimentally, the proposed DKF using the proposed multiplication method and the proposed fast polynomial filter was evaluated. The DKF introduced by Olfati was experimentally tested as well. Figure 11 shows the comparison for both methods for different numbers of neighbors. The results show that the proposed DKF achieves up to 33% energy saving. The results show also that one node can run the Olfati's DKF for up to five neighbors only, but the proposed DKF can run for up to seven neighbors. This different in the nodes numbers is because of the memory limitation, as Olfati's DKF exchange the measurements and the covariance, but the proposed DKF exchange the estimation only. Moreover the proposed multiplication method saves memory as well.

10. Conclusion

We have presented a low-power distributed Kalman filter based on a fast polynomial filter. Fast convergence leaded to significant energy saving. In addition, we proposed a light-weight energy-efficient multiplication algorithm. The proposed multiplication method reduced the number of add operations during multiplication by rounding any sequence of 1's in the fractional part. The applied rounding reduced the number of instruction cycles and reduced the memory storage without increasing the code complexity. The experimental results show that the proposed DKF achieved up to 33% energy consumption save compared to Olfsti's DKF. Moreover, the proposed DKF efficiently uses the node's memory, so each node can run DKF with up to seven neighbors.

References

[1] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks: methods, models, and classifications," *ACM Computing Surveys*, vol. 39, no. 3, Article ID 1267073, 2007.

[2] Micaz datasheet, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf.

[3] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 364–369, April 2005.

[4] A. Abdelgawad, S. Abdelhak, S. Ghosh, and M. Bayoumi, "A low-power multiplication algorithm for signal processing in wireless sensor networks," in *Proceedings of the 52nd IEEE*

International Midwest Symposium on Circuits and Systems (MWSCAS '09), pp. 695–698, Cancun, Mexico, August 2009.

[5] Y. Hatano and M. Mesbahi, "Agreement over random networks," in *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC '04)*, pp. 2010–2015, December 2004.

[6] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," in *Proceedings of the 42nd IEEE Conference on Decision and Control (CDC '03)*, vol. 5, pp. 4997–5002, December 2003.

[7] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[8] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005.

[9] L. Xiao, S. Boyd, and S. Lall, "A Scheme for robust distributed sensor fusion based on average consensus," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 63–70, April 2005.

[10] S. Kar and J. M. F. Moura, "Distributed average consensus in sensor networks with quantized inter-sensor communication," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '08)*, pp. 2281–2284, March-April 2008.

[11] S. Kirti and A. Scaglione, "Scalable distributed Kalman filtering through consensus," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '08)*, pp. 2725–2728, March-April 2008.

[12] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC '05)*, pp. 6698–6703, December 2005.

[13] R. Olfati Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proceedings of American Control Conference*, pp. 951–956, June 2003.

[14] R. Olfati-Saber, "Distributed tracking for mobile sensor networks with information-driven mobility," in *Proceedings of the American Control Conference (ACC '07)*, pp. 4606–4612, July 2007.

[15] G. Scutari, S. Barbarossa, and L. Pescosolido, "Distributed decision through self-synchronizing sensor networks in the presence of propagation delays and nonreciprocal channels," in *Proceedings of the 8th IEEE Signal Processing Advances in Wireless Communications (SPAWC '07)*, pp. 1–5, June 2007.

[16] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proceedings of the 46th IEEE Conference on Decision and Control (CDC '07)*, pp. 5492–5498, December 2007.

[17] L. Xiao, S. Boyd, and S. J. Kim, "Distributed average consensus with least-mean-square deviation," in *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS '06)*, pp. 2768–2776, July 2006.

[18] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links—part I: distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[19] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '06)*, pp. 635–639, July 2006.

[20] H. Chen and X. R. Li, "On track fusion with communication constraints," in *Proceedings of the 10th International Conference on Information Fusion (FUSION '07)*, pp. 1–7, July 2007.

[21] A. Ribeiro, G. B. Giannakis, and S. I. Roumeliotis, "SOI-KF: distributed Kalman filtering with low-cost communications using the sign of innovations," *IEEE Transactions on Signal Processing*, vol. 54, no. 12, pp. 4782–4795, 2006.

[22] R. Olfati-Saber and N. F. Sandell, "Distributed tracking in sensor networks with limited sensing range," in *Proceedings of American Control Conference (ACC '08)*, pp. 3157–3162, June 2008.

[23] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC '05)*, pp. 8179–8184, December 2005.

[24] A. Speranzon, C. Fischione, B. Johansson, and K. H. Johansson, "Adaptive distributed estimation over wireless sensor networks with packet losses," in *Proceedings of the 46th IEEE Conference on Decision and Control (CDC '07)*, pp. 5472–5477, December 2007.

[25] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed Kalman filtering based on consensus strategies," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, Article ID 4497788, pp. 622–633, 2008.

[26] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Approximate distributed Kalman filtering in sensor networks with quantifiable performance," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 133–139, April 2005.

[27] U. A. Khan and J. M. F. Moura, "Distributing the Kalman filter for large-scale systems," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4919–4935, 2008.

[28] U. A. Khan and J. M. F. Moura, "Model distribution for distributed Kalman filters: a graph theoretic approach," in *Proceedings of the 41st Asilomar Conference on Signals, Systems and Computers (ACSSC '07)*, pp. 611–615, November 2007.

[29] S. M. Azizi and K. Khorasani, "A distributed Kalman filter for actuator fault estimation of deep space formation flying satellites," in *Proceedings of the 3rd IEEE International Systems Conference*, pp. 354–359, March 2009.

[30] J. Dong, Q. Chen, and Z. Niu, "Random graph theory based connectivity analysis in wireless sensor networks with Rayleigh fading channels," in *Proceedings of the Asia-Pacific Conference on Communications (APCC '07)*, pp. 123–126, October 2007.

[31] B. Bollabas, *Random Graphs*, Cambride University Press, Cambride, UK, 2nd edition, 2001.

[32] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME, Journal of Basic Engineering*, pp. 35–45, 1960.

[33] S. C. Weller and N. C. Mann, "Assessing rater performance without a 'gold standard' using consensus theory," *Medical Decision Making*, vol. 17, no. 1, pp. 71–79, 1997.

[34] J. N. Tsitsiklis and M. Athans, "Convergence and asymptotic agreement in distributed decision problems," *IEEE Transactions on Automatic Control*, vol. 29, no. 1, pp. 42–50, 1984.

[35] E. Kokiopoulou and P. Frossard, "Polynomial filtering for fast convergence in distributed consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 342–354, 2009.

[36] S. Kar and J. M. F. Moura, "Sensor networks with random links: topology design for distributed consensus," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3315–3326, 2008.

[37] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 307–320, November 2006.